A Wallace(Dadda) multiplier generator.

Hiro Mori
Dec. 2,2023
bqe10133@gmail.com
X(twitter): ubukuproject

SUMMARY

The program generates Wallace(Dadda) multiplier. The outputs are netlist,critical path,layout and so on. The below 8bx8b example shows how to use the program.

0 command(8bx8b multiplier example)

Put five files
(script_multiplier,tree.cpp,cla4b_template,layout_template,multiplier_tb_template) into a new directory(any name is OK). Under the directory,

% chmod 755 script_multiplier
% ./script_multiplier 8

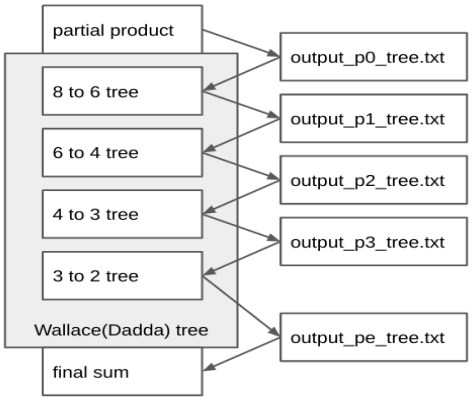
If you change 8 to 16, 16bx16b multiplier is generated.

1 8bx8b multiplier example

1.1 multiplier block,output files,programs

block	output files	programs
input: a[7:0],b[7:0]	output_p0_ab.txt	script_multiplier
partial product	output_p0_tree.txt	script_multiplier
Wallace(Dadda) tree	output_p1_tree.txt output_p2_tree.txt output_p3_tree.txt output_pe_tree.txt	script_multiplier tree.cpp
final sum: s[15:0]	output_p0_finalsum.txt output_p1_finalsum.txt output_p2_finalsum.txt output_p3_finalsum.txt	script_multiplier cla4b_template

[simulation] outputfiles:**multiplier_tb.cpp,output_netlist.txt,simulation.log,**
output_critical.txt,output_criticalpath.txt
programs:**script_multiplier,multiplier_tb_template**
[layout] outputfiles:**output_layout.txt,layout.html**
programs:**script_multiplier,layout_template**



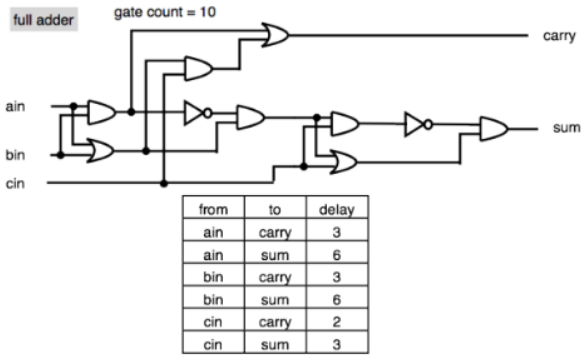
(example) part of output_p2_tree.txt

node,layout(x),layout(y),delay,function,input nodes

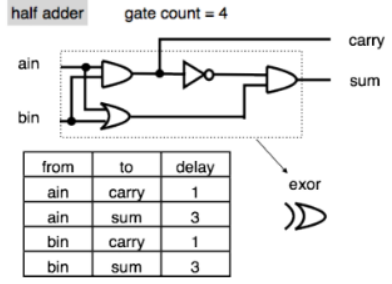
---

p2_5_0 100 380 2000 sim_ha_cout p1_4_0 p1_4_1
p2_5_1 100 400 7000 sim_fa_sout p1_5_2 p1_5_1 p1_5_0
p2_5_2 100 420 4000 sim_ha_sout p1_5_3 p1_5_4
p2_5_3 100 440 1000 sim_wire0 p1_5_5
p2_6_0 120 380 7000 sim_fa_sout p1_6_0 p1_6_5 p1_6_4
p2_6_1 120 400 4000 sim_fa_cout p1_5_2 p1_5_1 p1_5_0
p2_6_2 120 420 2000 sim_ha_cout p1_5_3 p1_5_4

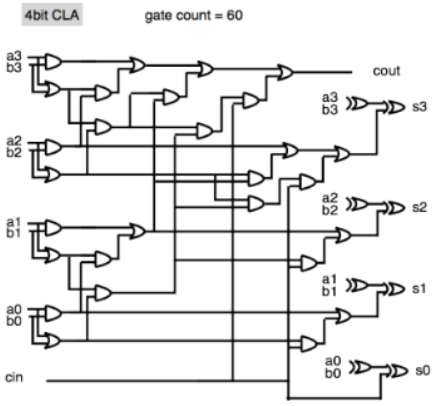
1.2 full adder,half adder,4b CLA adder



from	to	delay
ain	carry	3
ain	sum	6
bin	carry	3
bin	sum	6
cin	carry	2
cin	sum	3



from	to	delay
ain	carry	1
ain	sum	3
bin	carry	1
bin	sum	3



	cout	s3	s2	s1	s0
a3,b3	5	6	-	-	-
a2,b2	5	7	6	-	-
a1,b1	6	9	7	6	-
a0,b0	6	9	7	6	6
cin	2	5	5	5	3

2 program(C program,shell script,javascript)

All files link(google drive).
https://drive.google.com/file/d/1hT78J8v7-0_09lJUtp3pfxMLnX-i2Srr/view?usp=drive_link

[script_multiplier]
(step 0) removes old *.o,output_p*.txt.
(step 1) generates multiplier input a,b.
(step 2) generates partial product.
(step 3) generates Wallce/Dadda tree. The last tree is output_pe_tree.txt.
(step 4) generates final sum adder. CLA 4b is used.
(step 5) generates the whole netlist(step1-4).
(step 6) executes logic/delay simulation.
(step 7) extracts the critical path.
(step 8) generates layout.html.

% chmod 755 script_multiplier
% ./script_multiplier 8
8 x 8 wallace(dadda) multiplier
(step 0) file(old *.o,output_p*.txt) is removed.
(step 1) file(output_p0_ab.txt) is generated.
(step 2) file(output_p0_tree.txt) is generated.
8 to 6 tree
6 to 4 tree
4 to 3 tree
3 to 2 tree
(step 3) file(output_p*_tree.txt) is generated.
(step 4) file(output_p*_finalsum.txt) is generated.
(step 5) file(output_netlist.txt) is generated.
(step 6) file(multiplier_tb.o) is generated.

PASS num is 10000
FAIL num is 0

logic simulation is done. simulation.log is generated.

sum delay s[0]:7000 s[1]:7000 s[2]:10000 s[3]:13000
sum delay s[4]:16000 s[5]:19000 s[6]:20000 s[7]:22000
sum delay s[8]:24000 s[9]:26000 s[10]:26000 s[11]:26000
sum delay s[12]:26000 s[13]:28000 s[14]:28000 s[15]:28000

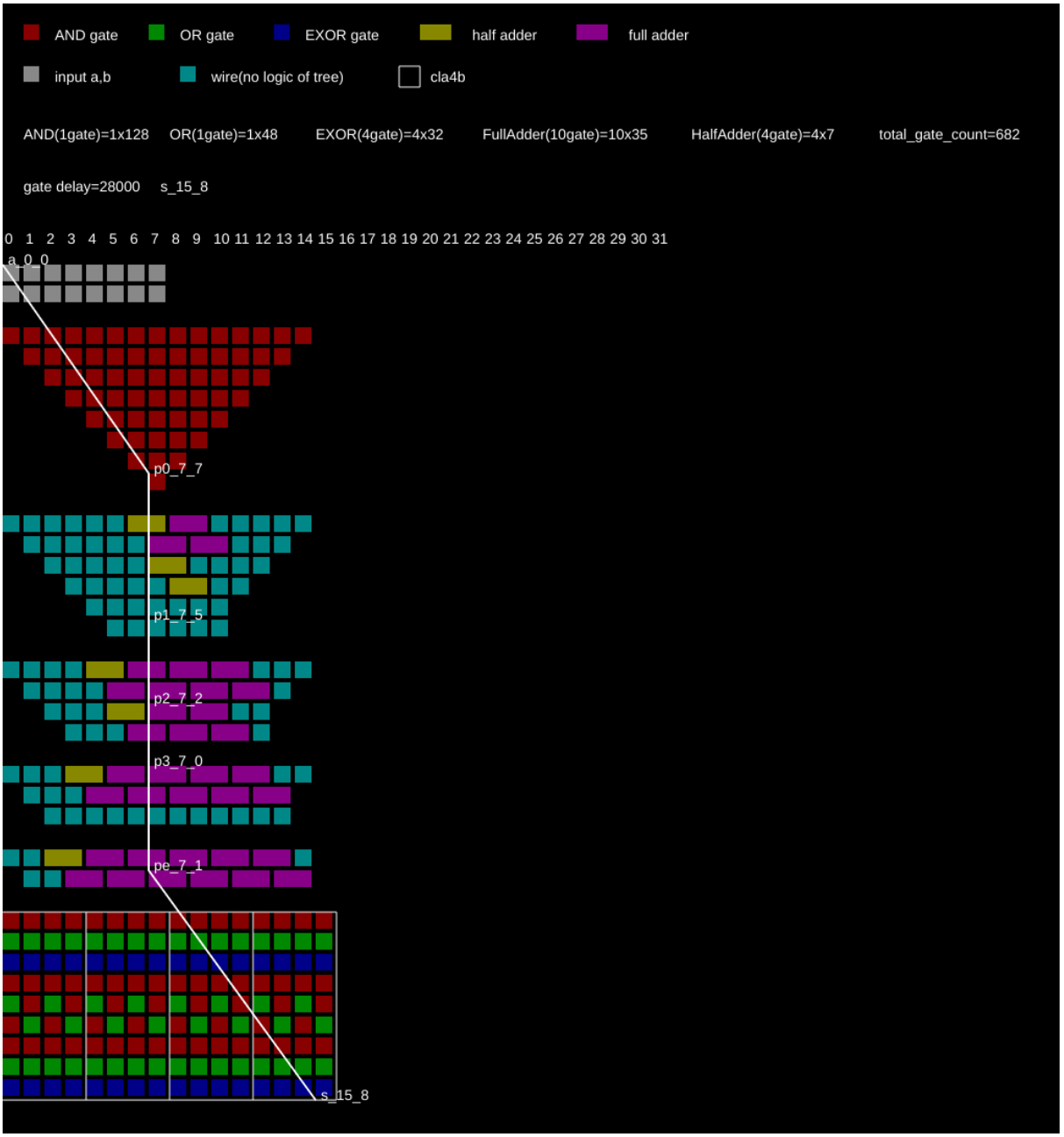
critical path is... (delay:AND=OR=INV=1000)
delay signal name
28000 s_15_8
25000 f_14_7
24000 f_14_6
23000 f_11_7
22000 f_11_6
21000 f_7_7
20000 f_7_6
19000 f_7_4
18000 f_6_3
17000 f_7_1
16000 pe_7_1
12000 p3_7_0
7000 p2_7_2
1000 p1_7_5
1000 p0_7_7
0 a_0_0
0 b_7_1

(step 7) file(output_critical.txt,output_criticalpath.txt) is generated.

gate count is...
AND(1gate) 128
OR(1gate) 48
EXOR(4gate) 32
Full Adder(10gate) 35
Half Adder(4gate) 7
gate_total 682

(step 8) file(layout.html) is generated.

layout.html



[release history]
1st release June 24,2023 : all codes from scratch
2nd release June 25,2023
3rd release July 2,2023 : layout.html modified
4th release July 9,2023 : all files modified
5th release July 10,2023 : gate count added
6th release July 12,2023 : layout_template2
7th release July 15,2023 : all files modified
8th release July 16,2023 : output_p*_finalsum.txt modified
9th release July 18,2023 : layout_template modified
10th release July 20,2023 : version header 1.0
11th release July 24,2023 : output_p*.txt format changed version 1.1

script_multiplier

```
#--------------------------------------------------------------------------------
# version 1.1 July 24,2023
#--------------------------------------------------------------------------------
echo "$1 x $1 wallace(dadda) multiplier"
#--------------------------------------------------------------------------------
# (step 0) remove old *.o,output_p*.txt
#--------------------------------------------------------------------------------
rm -f *.o output_p*.txt

echo '(step 0) file(old *.o,output_p*.txt) is removed.'

#--------------------------------------------------------------------------------
# (step 1) input a,b
#--------------------------------------------------------------------------------
width=`echo "$1"|awk '{print $1}'`
offsety=0

echo -n > output_p0_ab.txt
for ((i=0;i<$1;i++))
do
  echo "$i"|awk '{print "a_" $1 "_0 0 0 0 NONE NONE"}' >> output_p0_ab.txt
done
for ((i=0;i<$1;i++))
do
  echo "$i"|awk '{print "b_" $1 "_1 0 0 0 NONE NONE"}' >> output_p0_ab.txt
done

#layout
sed 's/_/ /' output_p0_ab.txt|sed 's/_/ /'|awk '{print $1,$2,$3,$2*VAR0,$3*VAR0+VAR1,$6,$7,$8,$9,$10}'
VAR0=20 VAR1=$offsety > tmp_layout0000
awk '{print $1 "_" $2 "_" $3,$4,$5,$6,$7,$8,$9,$10}' tmp_layout0000 > output_p0_ab.txt
offsety=`echo '2'|awk '{print $1*VAR+20}' VAR=20`

echo '(step 1) file(output_p0_ab.txt) is generated.'

#--------------------------------------------------------------------------------
# (step 2) partial product
#--------------------------------------------------------------------------------
length=`echo "$1"|awk '{print $1*2}'`

echo -n > tmp_pp0000
for ((i=0;i<$1;i++))
do
  for ((j=0;j<$1;j++))
  do
    echo "$j $i"|awk '{print $1+$2}'|awk '{print "p0",$1,VAR0,"0 0 1000 sim_and a_" VAR1 "_0 b_" VAR0 "_1"}'
VAR0=$i VAR1=$j >> tmp_pp0000
```

```
  done
done

echo -n > output_p0_tree.txt
for ((i=0;i<$length;i++))
do
  awk '{if($2==VAR)print $0}' VAR=$i tmp_pp0000|cat -n|awk '{print $2 "_" $3 "_" $1-1,$5,$6,$7,$8,$9,$10,$11}'
>> output_p0_tree.txt
done

#layout
sed 's/_/ /' output_p0_tree.txt|sed 's/_/ /'|awk '{print $1,$2,$3,$2*VAR0,$3*VAR0+VAR1,$6,$7,$8,$9,$10}'
VAR0=20 VAR1=$offsety > tmp_layout0000
awk '{print $1 "_" $2 "_" $3,$4,$5,$6,$7,$8,$9,$10}' tmp_layout0000 > output_p0_tree.txt
offsety=`echo "$offsety $width"|awk '{print $1+$2*VAR+20}' VAR=20`

echo '(step 2) file(output_p0_tree.txt) is generated.'
rm tmp_pp*

#-----------------------------------------
# (step3) tree
#-----------------------------------------
m=0
g++ -o tree.o tree.cpp
while true
do
  width1=`sed 's/_/ /' output_p"$m"_tree.txt|sed 's/_/ /'|awk '{print $3}'|sort -nr |uniq|head -1|awk '{print $1+1}'`
  width2=`echo "$width1"|awk '{print $1-int($1/3)}'`
  echo "$width1 to $width2 tree"
  n=`echo "$m"|awk '{print $1+1}'`
  echo 'Z' > tmp_tree0000

  #---------------------------------------
  #tree.cpp loop
  #tmp_tree0000:y for the upper bit
  #tmp_tree0001:y of the current bit
  #---------------------------------------
  for ((i=0;i<$length;i++))
  do
    sed 's/_/ /' output_p"$m"_tree.txt|sed 's/_/ /'|awk '{if($2==VAR)print $3,$6}' VAR=$i|sort -n|awk '{print $2,$1}'
> tmp_tree0001
    echo 'Z' >> tmp_tree0001
    ./tree.o tmp_tree0001 $width2 $i $m tmp_tree0000 > tmp_tree0002_"$m"_"$i"
    grep table tmp_tree0002_"$m"_"$i"|grep _cout|awk '{print $6}' > tmp_tree0000
    echo 'Z' >> tmp_tree0000
  done

  grep table tmp_tree0002_"$m"_*|awk '{print $2,$4,$6,"0 0",$8,$9,$10,$11,$12}'|sort -n|awk '{print $1 "_" $2
"_" $3,$4,$5,$6,$7,$8,$9,$10}' > output_p"$n"_tree.txt
  m=`echo "$m"|awk '{print $1+1}'` #m++

  if [ "$width2" != 2 ]; then
    #layout
    sed 's/_/ /' output_p"$n"_tree.txt|sed 's/_/ /'|awk '{print $1,$2,$3,$2*VAR0,$3*VAR0+VAR1,$6,$7,$8,$9,$10}'
VAR0=20 VAR1=$offsety > tmp_layout0000
    awk '{print $1 "_" $2 "_" $3,$4,$5,$6,$7,$8,$9,$10}' tmp_layout0000 > output_p"$n"_tree.txt
    offsety=`echo "$offsety $width2"|awk '{print $1+$2*VAR+20}' VAR=20`
  fi

  if [ "$width2" = 2 ]; then
    #---------------------------------------
    # the last tree:go to the final sum
    #---------------------------------------
    #padding input
    sed 's/_/ /' output_p"$n"_tree.txt|sed 's/_/ /' > tmp_tree0003
    for ((i=0;i<$length;i++))
```

```
    do
      for ((j=0;j<2;j++))
      do
        hit=`grep "p$n $i $j" tmp_tree0003|wc -l|awk '{print $1}'`
        echo "$hit"|awk '{if($1==0)print "p" VAR2,VAR0,VAR1,"0 0 0 sim_wire1 0"}' VAR0=$i VAR1=$j VAR2=$n
>> tmp_tree0003
      done
    done
    sort -n tmp_tree0003|awk '{print $1 "_" $2 "_" $3,$4,$5,$6,$7,$8,$9,$10}' > output_p"$n"_tree.txt
    cat output_p"$n"_tree.txt|sed "s/p$n/pe/g" >  output_pe_tree.txt
    rm output_p"$n"_tree.txt

    #layout
    sed 's/_/ /' output_pe_tree.txt|sed 's/_/ /'|awk '{print $1,$2,$3,$2*VAR0,$3*VAR0+VAR1,$6,$7,$8,$9,$10}'
VAR0=20 VAR1=$offsety > tmp_layout0000
    awk '{print $1 "_" $2 "_" $3,$4,$5,$6,$7,$8,$9,$10}' tmp_layout0000 > output_pe_tree.txt
    offsety=`echo "$offsety $width2"|awk '{print $1+$2*VAR+20}' VAR=20`

    break
  fi

done

echo '(step 3) file(output_p*_tree.txt) is generated.'
rm tmp_tree*

#---------------------------------------------
# (step4) finalsum
#---------------------------------------------
for ((i=0;i<($length/4);i++))
do
  #---------------------
  #4bit CLA
  #---------------------
  offset0=`echo "$i"|awk '{print $1*4+0}'`
  offset1=`echo "$i"|awk '{print $1*4+1}'`
  offset2=`echo "$i"|awk '{print $1*4+2}'`
  offset3=`echo "$i"|awk '{print $1*4+3}'`
  offset4=`echo "$i"|awk '{print $1*4-1}'`
  grep -v ^# cla4b_template|sed "s/BIT0/$offset0/g"|sed "s/BIT1/$offset1/g"|sed "s/BIT2/$offset2/g"|sed
"s/BIT3/$offset3/g"|sed "s/BIT4/$offset4/g"|
  sed 's/f_\-1_7/fzero/g' > output_p"$i"_finalsum.txt

  #layout
  sed 's/_/ /' output_p"$i"_finalsum.txt|sed 's/_/ /'|awk '{print
$1,$2,$3,$2*VAR0,$3*VAR0+VAR1,$6,$7,$8,$9,$10}' VAR0=20 VAR1=$offsety > tmp_layout0000
  awk '{print $1 "_" $2 "_" $3,$4,$5,$6,$7,$8,$9,$10}' tmp_layout0000 > output_p"$i"_finalsum.txt
done

echo '(step 4) file(output_p*_finalsum.txt) is generated.'

#---------------------------------------------
# (step 5) netlist
#---------------------------------------------
# input bridge a[0]->a_0_0
echo -n > tmp_netlist1000
for ((i=0;i<$width;i++))
do
  echo "$i"|awk '{print "int a_" $1 "_0;"}' >> tmp_netlist1000
  echo "$i"|awk '{print "int b_" $1 "_1;"}' >> tmp_netlist1000
done
for ((i=0;i<$width;i++))
do
  echo "$i"|awk '{print "a_" $1 "_0=a[" $1 "];"}' >> tmp_netlist1000
  echo "$i"|awk '{print "b_" $1 "_1=b[" $1 "];"}' >> tmp_netlist1000
done
```

```
# output bridge s_0_8->s[0]
echo -n > tmp_netlist1001
for ((i=0;i<$length;i++))
do
  echo "$i"|awk '{print "s[" $1 "]=s_" $1 "_8;"}' >> tmp_netlist1001
done

# netlist(file order is important)
echo -n > tmp_netlist2000
for ((i=0;i<$m;i++))
do
  cat output_p"$i"_tree.txt >> tmp_netlist2000
done
cat output_pe_tree.txt >> tmp_netlist2000
for ((i=0;i<($length/4);i++))
do
  cat output_p"$i"_finalsum.txt >> tmp_netlist2000
done
awk '{print "int",$1 ";"}' tmp_netlist2000 > tmp_netlist2001
awk '{print $1 "=" $5 "(" $6 "," $7 "," $8 ",delay);"}' tmp_netlist2000|sed 's/,,,/,/g'|sed 's/,,/,/g' >> tmp_netlist2001
echo 'void netlist(int *a,int *b,int *s,int delay){' > output_netlist.txt
echo 'int fzero=0;' >> output_netlist.txt
cat tmp_netlist1000 tmp_netlist2001 tmp_netlist1001 >> output_netlist.txt
echo '  if(delay==1){' >> output_netlist.txt
grep = tmp_netlist2001|sed 's/=/ /g'|awk '{print "     printf(\"" "gate_delay " $1, "%d\\n\",",$1 ");"}' >>
output_netlist.txt
echo '}}' >> output_netlist.txt

echo '(step 5) file(output_netlist.txt) is generated.'
rm tmp_netlist*

#--------------------------------------------
# (step 6) simulation
#--------------------------------------------
sed "s/BITSIZE0/$width/g" multiplier_tb_template > tmp_simulation0000
cat tmp_simulation0000 output_netlist.txt > multiplier_tb.cpp

g++ -o multiplier_tb.o multiplier_tb.cpp
./multiplier_tb.o > simulation.log

echo '(step 6) file(multiplier_tb.o) is generated.'
echo ''
grep PASS simulation.log|wc -l|awk '{print "PASS num is " $1}'
grep FAIL simulation.log|wc -l|awk '{print "FAIL num is " $1}'
echo ''
echo 'logic simulation is done. simulation.log is generated.'
echo ''
grep 'sum delay' simulation.log

rm tmp_simulation*

#--------------------------------------------
# (step 7) critical path
#--------------------------------------------
#search file
grep gate_delay simulation.log|awk '{print $3,$2}' > tmp_critical0000
grep sim_ output_netlist.txt|sed 's/=/ /g'|sed 's/(/ /g'|sed 's/)/ /g'|sed 's/\,/ /g'|sed 's/\;/ /g'|sed 's/delay//g'|sed 's/ 0/
NONE/g'|awk '{print $0,"NONE NONE"}' > tmp_critical0001
cp tmp_critical0000 tmp_critical0002

echo -n > output_critical.txt
while true
do
  node=`sort -nr tmp_critical0002|head -1|awk '{print $2}'`
  echo -n "p $node " >> output_critical.txt
```

```
  grep "$node" tmp_critical0000|awk '{print $1}' >> output_critical.txt
  send=`grep "$node" tmp_critical0000|awk '{print $1}'`
  send2=`grep "$node" tmp_critical0000|grep 'p0'|wc -l|awk '{print $1}'`
  node0=`grep "$node" tmp_critical0001|awk '{if($1==VAR)print $0}' VAR=$node|awk '{print $3}'`
  node1=`grep "$node" tmp_critical0001|awk '{if($1==VAR)print $0}' VAR=$node|awk '{print $4}'`
  node2=`grep "$node" tmp_critical0001|awk '{if($1==VAR)print $0}' VAR=$node|awk '{print $5}'`
  echo "c $node0" >> output_critical.txt
  echo "c $node1" >> output_critical.txt
  echo "c $node2" >> output_critical.txt
  grep "$node0" tmp_critical0000 > tmp_critical1000
  grep "$node1" tmp_critical0000 >> tmp_critical1000
  grep "$node2" tmp_critical0000 >> tmp_critical1000
  cp tmp_critical1000 tmp_critical0002

  if [ "$send" = 1000 ]; then
    if [ "$send2" = 1 ]; then
      break
    fi
  fi
done

awk '{if($1=="p")print $3,$2}' output_critical.txt > output_criticalpath.txt
egrep 'a_|b_' output_critical.txt|awk '{print "0", $2}' >> output_criticalpath.txt

echo ''
echo 'critical path is... (delay:AND=OR=INV=1000)'
echo 'delay signal name'
cat output_criticalpath.txt|awk '{print $1,$2}'

echo ''
echo '(step 7) file(output_critical.txt,output_criticalpath.txt) is generated.'
rm tmp_critical*

#-------------------------------------------
# (step 8) layout
#-------------------------------------------
# ab,p0,p1,,,
offset=250
cat output_p0_ab.txt output_p*_tree.txt output_p*_finalsum.txt > output_layout.txt
awk '{print $2,$3,$5}' output_layout.txt|
sed 's/NONE/#888888/g'|sed 's/sim_and/#880000/g'|sed 's/sim_wire0/#008888/g'|sed
's/sim_wire1/#000000/g'|sed 's/0;/#000000/g'|
sed 's/sim_or/#008800/g'|sed 's/sim_exor/#000088/g'|
sed 's/sim_ha_sout/#FFFFF0/g'|sed 's/sim_ha_cout/#888800/g'|sed 's/sim_fa_sout/#FFFFF1/g'|sed
's/sim_fa_cout/#880088/g'  > tmp_layout3000
awk '{print "ctx.fillStyle=\"" $3 "\";","ctx.fillRect(" $1 "," $2+VAR1 "," VAR0 "," VAR0 ");"}' VAR0=16
VAR1=$offset tmp_layout3000 > output_layout.txt

#fa,ha modification
egrep 'FFFFF0|FFFFF1' output_layout.txt|sed 's/16\,16/20\,16/g'> tmp_layout5000
sed 's/FFFFF0/888800/g' tmp_layout5000|sed 's/FFFFF1/880088/g'> tmp_layout5001
grep -v 'FFFFF0' output_layout.txt |grep -v 'FFFF1'> tmp_layout5002
cat tmp_layout5001 tmp_layout5002 > output_layout.txt

#finalsum rectangle
for ((i=0;i<($length/4);i++))
do
  echo 'ctx.beginPath();' >>output_layout.txt
  echo "$i"|awk '{print "ctx.moveTo(" ($1+0)*4*VAR2 "," VAR0+VAR1      ");"}' VAR0=$offset VAR1=$offsety
VAR2=20 >>output_layout.txt
  echo "$i"|awk '{print "ctx.lineTo(" ($1+1)*4*VAR2 "," VAR0+VAR1      ");"}' VAR0=$offset VAR1=$offsety
VAR2=20 >>output_layout.txt
  echo "$i"|awk '{print "ctx.lineTo(" ($1+1)*4*VAR2 "," VAR0+VAR1+180 ");"}' VAR0=$offset VAR1=$offsety
VAR2=20 >>output_layout.txt
  echo "$i"|awk '{print "ctx.lineTo(" ($1+0)*4*VAR2 "," VAR0+VAR1+180 ");"}' VAR0=$offset VAR1=$offsety
VAR2=20 >>output_layout.txt
```

```
  echo 'ctx.strokeStyle="white";' >>output_layout.txt
  echo 'ctx.lineWidth=1;' >>output_layout.txt
  echo 'ctx.stroke();' >>output_layout.txt
done

#gate count
echo ''
echo 'gate count is...'
gate_and=`grep sim_and output_netlist.txt|wc -l|awk '{print $1}'`
echo "AND(1gate) $gate_and"
gate_or=`grep sim_or output_netlist.txt|wc -l|awk '{print $1}'`
echo "OR(1gate)  $gate_or"
gate_exor=`grep sim_exor output_netlist.txt|wc -l|awk '{print $1}'`
echo "EXOR(4gate)  $gate_exor"
gate_fa=`grep sim_fa output_netlist.txt|wc -l|awk '{print $1/2}'`
echo "Full Adder(10gate)  $gate_fa"
gate_ha=`grep sim_ha output_netlist.txt|wc -l|awk '{print $1/2}'`
echo "Half Adder(4gate)  $gate_ha"
echo "ctx.font='14px Helvetica';" >>output_layout.txt
echo 'ctx.fillStyle="#ffffff";' >>output_layout.txt

echo "AND(1gate)  $gate_and" |awk '{print "ctx.fillText(\"" $1 "=1x" $2 "\", 20,130);"}' >>output_layout.txt
echo "OR(1gate)   $gate_or"  |awk '{print "ctx.fillText(\"" $1 "=1x" $2 "\",160,130);"}' >>output_layout.txt
echo "EXOR(4gate) $gate_exor"|awk '{print "ctx.fillText(\"" $1 "=4x" $2 "\",300,130);"}' >>output_layout.txt
echo "FullAdder(10gate) $gate_fa"|awk '{print "ctx.fillText(\"" $1 "=10x" $2 "\",460,130);"}' >>output_layout.txt
echo "HalfAdder(4gate) $gate_ha"|awk '{print "ctx.fillText(\"" $1 "=4x" $2 "\",660,130);"}' >>output_layout.txt
gate_total=`echo "$gate_and $gate_or $gate_exor $gate_fa $gate_ha"|awk '{print
"gate_total",$1+$2+$3*4+$4*10+$5*4}'`
echo "$gate_total"
echo ''
echo "total_gate_count  $gate_total" |awk '{print "ctx.fillText(\"" $1 "=" $3 "\", 840,130);"}' >>output_layout.txt

#critical path line
echo 'ctx.beginPath();' >>output_layout.txt
snode=`grep ^p output_critical.txt|head -1|awk '{print $2}'`
sx=`grep ^p output_critical.txt|head -1|awk '{print $2}'|sed 's/s_//g'|sed 's/_8//g'|awk '{print $1*VAR}' VAR=20`
echo "$sx $offsety $snode" > tmp_layout4000
awk '{print "ctx.moveTo(" $1 "," $2+VAR+180 ");"}' VAR=$offset tmp_layout4000 >>output_layout.txt
awk '{print $2,$1}' output_criticalpath.txt|egrep '^p|^a' > tmp_layout6000
awk '{print "grep \"^" $1 "\" output_p*.txt"}' tmp_layout6000 > tmp_layout6001
chmod 755 tmp_layout6001
./tmp_layout6001 > tmp_layout6002
awk '{print "ctx.lineTo(" $2 "," $3+VAR ");"}' VAR=$offset tmp_layout6002 >>output_layout.txt
echo 'ctx.strokeStyle="white";' >>output_layout.txt
echo 'ctx.lineWidth=2;' >>output_layout.txt
echo 'ctx.stroke();' >>output_layout.txt

#critical path signal name
echo "ctx.font='14px Helvetica';" >>output_layout.txt
echo 'ctx.fillStyle="#ffffff";' >>output_layout.txt
awk '{print "ctx.fillText(\"" $3 "\"," $1+5 "," $2+VAR+180 ");"}' VAR=$offset tmp_layout4000 >>output_layout.txt
sed 's/:/ /g' tmp_layout6002|awk '{print "ctx.fillText(\"" $2 "\"," $3+5 "," $4+VAR ");"}' VAR=$offset
>>output_layout.txt
head -1 output_criticalpath.txt|awk '{print "ctx.fillText(\"gate delay=" $1,"   ",$2 "\", 20,180);"}'
>>output_layout.txt

#layout.html
sed -n '1,109p' layout_template > tmp_layout7000
sed -n '110,125p' layout_template > tmp_layout7001
cat tmp_layout7000 output_layout.txt tmp_layout7001 > layout.html

echo '(step 8) file(layout.html) is generated.'
rm tmp_layout*
```

tree.cpp

```
//--------------------------------------------------------------------------------
// version 1.1 July 24,2023
//--------------------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <math.h>
using namespace std;
void treecalc(int *din,int *dinnum,int dinlen,int afternum,int x,int z,int carrynum,int *carryin);
void treesort(int *din,int *dinnum,int dinlen,int reverse);

//---------------------------------
//function : main
//---------------------------------
int main(int argc, char * const argv[]) {
  int  mainbuf[2048];//x data
  int  mainbuf2[2048];//carry info
  char c0;
  int  d0;
  int  i,m;
  int  afternum;
  int  din[2048];
  int  dinnum[2048];
  int  dinlen;
  int  dinvalue;
  int  dinflag;
  int  carryvalue;
  int  carryflag;
  int  x;
  int  z;
  int  carrynum;
  int  carryin[2048];

  ifstream fv0(argv[1]);//x data
  afternum = atoi(argv[2]);
  x        = atoi(argv[3]);
  z        = atoi(argv[4]);
  ifstream fv1(argv[5]);//carry info

  //x data
  for(i=0;i<2048;i++){
    mainbuf[i]=0;
  }
  for(i=0;i<2048;i++){
    fv0.get(c0);
    d0=(int)c0;
    mainbuf[i]=d0;
    if(d0==90){break;}
  }

  //carry info
  for(i=0;i<2048;i++){
    mainbuf2[i]=0;
  }
  for(i=0;i<2048;i++){
    fv1.get(c0);
    d0=(int)c0;
    mainbuf2[i]=d0;
    if(d0==90){break;}
  }

  //------------------------------
  // x data store
```

```
  //------------------------------
  m=0;
  dinvalue=0;
  dinflag=0;
  for(i=0;i<2048;i++){//enough big
    if(mainbuf[i]==90){//"Z" = 90
      break;
    }
    else if(mainbuf[i]==32 | mainbuf[i]==10){//" " = 32 or "CR"=10
      if(dinflag==1){
        if(m%2==0){
          din[m/2]=dinvalue;
        }
        else{
          dinnum[m/2]=dinvalue;
        }
        m++;
        dinvalue=0;
        dinflag=0;
      }
    }
    else{
      dinvalue=dinvalue*10;
      dinvalue=dinvalue+(mainbuf[i]-48);//"0" = 48
      dinflag=1;
    }
  }
  dinlen=m/2;

  //------------------------------
  // carry info store
  //------------------------------
  m=0;
  carryvalue=0;
  carryflag=0;
  for(i=0;i<2048;i++){//enough big
    if(mainbuf2[i]==90){//"Z" = 90
      break;
    }
    else if(mainbuf2[i]==32 | mainbuf2[i]==10){//" " = 32 or "CR"=10
      if(carryflag==1){
        carryin[m]=carryvalue;
        m++;
        carryvalue=0;
        carryflag=0;
      }
    }
    else{
      carryvalue=carryvalue*10;
      carryvalue=carryvalue+(mainbuf2[i]-48);//"0" = 48
      carryflag=1;
    }
  }
  carrynum=m;

  treecalc(din,dinnum,dinlen,afternum,x,z,carrynum,carryin);

  fv0.close();
  fv1.close();
  return 0;
}

//---------------------------------
//function : treecalc
//---------------------------------
void treecalc(int *din,int *dinnum,int dinlen,int afternum,int x,int z,int carrynum,int *carryin){
```

```
int i,j,k,m,n;
int flag;
int first_val,second_val,third_val;
int first_valnum,second_valnum,third_valnum;
int half_carry,half_sum;
int carrybuf[2048];
int sumbuf[2048];
int carrybufnum[2048];
int sumbufnum[2048];
int fanum,hanum,wirenum;
int first_carry,first_sum;
int second_carry,second_sum;
int afternum2;
int din2[2048];
int dinnum2[2048];
int dinlen2;
int numflag[2048];
int y_s;


for(i=0;i<afternum;i++){
  numflag[i]=0;
}

for(i=0;i<carrynum;i++){
  numflag[carryin[i]]=1;
}

afternum2=afternum-carrynum;

//------------------------------
// fanum,hanum,wirenum calculation
//------------------------------
if(dinlen <= afternum2){
  fanum  = 0;
  hanum  = 0;
  wirenum = dinlen;
}
else{
  fanum  = (dinlen-afternum2)/2;
  hanum  = (dinlen-afternum2)%2;
  wirenum = dinlen-3*fanum-2*hanum;
}
printf("------------------------------\n");
printf("dinlen:%d fanum:%d hanum:%d wirenum:%d\n",dinlen,fanum,hanum,wirenum);

//------------------------------
// din modification
//------------------------------
printf("================================\n");
printf("din modification\n");
printf("================================\n");

printf("----initial\n");

for(i=0;i<dinlen;i++){
  printf("%d ",din[i]);
}
printf("\n");

treesort(din,dinnum,dinlen,0);
printf("----sorted\n");

printf("din\n");
for(i=0;i<dinlen;i++){
  printf("%d ",din[i]);
```

```
}
printf("\n");

printf("dinnum\n");
for(i=0;i<dinlen;i++){
  printf("%d ",dinnum[i]);
}
printf("\n");

//------------------------------
// wire,ha,fa
//------------------------------
printf("================================\n");
printf("wire,ha,fa\n");
printf("================================\n");

  m=0;
  n=0;
  dinlen2=fanum*3;
  for(i=0;i<dinlen2;i++){
    din2[i]=1000*din[i];
    dinnum2[i]=dinnum[i];
  }

  //###################
  //full adder
  //###################
  for(i=0;i<fanum;i++){

    treesort(din2,dinnum2,dinlen2,1);

    flag=0;
    for(j=0;j<dinlen2;j++){
      if(j==0){
        din2[j]=din2[j]/1000;
        first_val=din2[j];
        first_valnum=dinnum2[j];
      }
      else if(j>0 & j<(fanum-i)){
        if((din2[j]/1000<=(din2[0]-3)) & flag==0){
          flag=1;
          din2[j]=din2[j]/1000;
          din2[j+1]=din2[j+1]/1000;
          second_val=din2[j];
          third_val=din2[j+1];
          second_valnum=dinnum2[j];
          third_valnum=dinnum2[j+1];
        }
      }
      else if(j==(fanum-i)){
        if(flag==0){
          if(din2[j]==din2[j-1]){
            flag=1;
            din2[j]=din2[j]/1000;
            din2[j-1]=din2[j-1]/1000;
            second_val=din2[j];
            third_val=din2[j-1];
            second_valnum=dinnum2[j];
            third_valnum=dinnum2[j-1];
          }
          else{
            flag=1;
            din2[j]=din2[j]/1000;
            din2[j+1]=din2[j+1]/1000;
            second_val=din2[j];
            third_val=din2[j+1];
```

```
            second_valnum=dinnum2[j];
            third_valnum=dinnum2[j+1];
          }
        }
      }
    }
    printf("----full adder %d %d %d    %d %d
%d\n",first_val,second_val,third_val,first_valnum,second_valnum,third_valnum);

    first_carry=first_val+2000;
    first_sum=first_val+3000;
    second_carry=second_val+3000;
    second_sum=second_val+6000;

    for(k=0;k<afternum;k++){
      if(numflag[k]==0){
        y_s=k;
        numflag[k]=1;
        break;
      }
    }

    if(first_sum>=second_sum){
      printf("table p%d   x %3d   y_s %3d  d %3d  sim_fa_sout p%d_%d_%d p%d_%d_%d p%d_%d_%d
\n",(z+1),x,y_s, first_sum,z,x,first_valnum,z,x,second_valnum,z,x,third_valnum);
      sumbuf[m]=first_sum;   m++;
    }
    else{
      printf("table p%d   x %3d   y_s %3d  d %3d  sim_fa_sout p%d_%d_%d p%d_%d_%d p%d_%d_%d
\n",(z+1),x,y_s,second_sum,z,x,first_valnum,z,x,second_valnum,z,x,third_valnum);
      sumbuf[m]=second_sum;   m++;
    }
    if(first_carry>=second_carry){
      printf("table p%d   x %3d   y_c %3d  d %3d  sim_fa_cout p%d_%d_%d p%d_%d_%d p%d_%d_%d
\n",(z+1),(x+1),y_s,first_carry,z,x,first_valnum,z,x,second_valnum,z,x,third_valnum);
      carrybuf[n]=first_carry; n++;
    }
    else{
      printf("table p%d   x %3d   y_c %3d  d %3d  sim_fa_cout p%d_%d_%d p%d_%d_%d p%d_%d_%d
\n",(z+1),(x+1),y_s,second_carry,z,x,first_valnum,z,x,second_valnum,z,x,third_valnum);
      carrybuf[n]=second_carry; n++;
    }
  }
  printf("\n");
  //###################
  //half adder
  //###################
  for(i=0;i<hanum;i++){

    for(k=0;k<afternum;k++){
      if(numflag[k]==0){
        y_s=k;
        numflag[k]=1;
        break;
      }
    }

    half_carry=din[fanum*3+1]+1000;
    half_sum  =din[fanum*3+1]+3000;
    sumbuf[m]=half_sum;   m++;
    carrybuf[n]=half_carry; n++;
    printf("----half adder %d %d   %d
%d\n",din[fanum*3],din[fanum*3+1],dinnum[fanum*3],dinnum[fanum*3+1]);
    printf("table p%d   x %3d   y_s %3d  d %3d  sim_ha_sout p%d_%d_%d p%d_%d_%d
\n",(z+1),x,y_s,half_sum,z,x,dinnum[fanum*3],z,x,dinnum[fanum*3+1]);
    printf("table p%d   x %3d   y_c %3d  d %3d  sim_ha_cout p%d_%d_%d p%d_%d_%d
```

```
\n",(z+1),(x+1),y_s,half_carry,z,x,dinnum[fanum*3],z,x,dinnum[fanum*3+1]);
    }
    //##################
    //wire
    //##################
    for(i=0;i<wirenum;i++){
       sumbuf[m]=din[fanum*3+hanum*2+i]; m++;
    }

     printf("----wire ");
    for(i=0;i<wirenum;i++){
      printf("%d ",din[fanum*3+hanum*2+i]);
    }
     printf("    ");
    for(i=0;i<wirenum;i++){
      printf("%d ",dinnum[fanum*3+hanum*2+i]);
    }
    printf("\n");
    for(i=0;i<wirenum;i++){

      for(k=0;k<afternum;k++){
        if(numflag[k]==0){
          y_s=k;
          numflag[k]=1;
          break;
        }
      }

      printf("table p%d   x %3d   y_s %3d  d %3d  sim_wire0 p%d_%d_%d
\n",(z+1),x,y_s,din[fanum*3+hanum*2+i],z,x,dinnum[fanum*3+hanum*2+i]);
    }
    //##################
    //summary
    //##################
    printf("============================\n");
    printf("summary\n");
    printf("============================\n");

    printf("----summary\n");
    printf("sum_num %d carry_num %d\n",m,n);

    printf("FINAL sumbuf ");
    printf("m:%d    ",m);
    for(i=0;i<m;i++){
      printf("%d ",sumbuf[i]);
    }
    printf("\n");

    printf("FINAL carrybuf ");
    printf("n:%d    ",n);
    if(n==0){
      printf("0");
    }
    else{
      for(i=0;i<n;i++){
         printf("%d ",carrybuf[i]);
      }
    }
    printf("\n\n");
}

//---------------------------------
//function : treesort
//---------------------------------
void treesort(int *din,int *dinnum,int dinlen,int reverse){
  int i,j,k;
```

```
  int min,tmp;
  int dintmp[2048];

  //-------------------------
  // sort body
  //-------------------------
  for(i=0;i<dinlen-1;i++){
    min=din[i];
    k=i;
    for(j=i+1;j<dinlen;j++){
      if(din[j]<min){
        min=din[j];
        k=j;
      }
    }

    tmp=din[i];
    din[i]=din[k];
    din[k]=tmp;
    tmp=dinnum[i];
    dinnum[i]=dinnum[k];
    dinnum[k]=tmp;
  }

  if(reverse==1){
    //sort reverse
    for(i=0;i<dinlen;i++){
      dintmp[dinlen-1-i]=din[i];
    }
    for(i=0;i<dinlen;i++){
      din[i]=dintmp[i];
    }
    for(i=0;i<dinlen;i++){
      dintmp[dinlen-1-i]=dinnum[i];
    }
    for(i=0;i<dinlen;i++){
      dinnum[i]=dintmp[i];
    }
  }
}
```

cla4b_template

```
#-------------------------------------------------------------------------------
# version 1.1 July 24,2023
#-------------------------------------------------------------------------------
f_BIT0_0 0 0 0 sim_and pe_BIT0_0 pe_BIT0_1
f_BIT1_0 0 0 0 sim_and pe_BIT1_0 pe_BIT1_1
f_BIT2_0 0 0 0 sim_and pe_BIT2_0 pe_BIT2_1
f_BIT3_0 0 0 0 sim_and pe_BIT3_0 pe_BIT3_1
f_BIT0_1 0 0 0 sim_or pe_BIT0_0 pe_BIT0_1
f_BIT1_1 0 0 0 sim_or pe_BIT1_0 pe_BIT1_1
f_BIT2_1 0 0 0 sim_or pe_BIT2_0 pe_BIT2_1
f_BIT3_1 0 0 0 sim_or pe_BIT3_0 pe_BIT3_1
f_BIT0_2 0 0 0 sim_exor pe_BIT0_0 pe_BIT0_1
f_BIT1_2 0 0 0 sim_exor pe_BIT1_0 pe_BIT1_1
f_BIT2_2 0 0 0 sim_exor pe_BIT2_0 pe_BIT2_1
f_BIT3_2 0 0 0 sim_exor pe_BIT3_0 pe_BIT3_1
f_BIT0_3 0 0 0 sim_and f_BIT0_1 f_BIT1_1
f_BIT1_3 0 0 0 sim_and f_BIT1_1 f_BIT0_0
f_BIT2_3 0 0 0 sim_and f_BIT2_1 f_BIT3_1
f_BIT3_3 0 0 0 sim_and f_BIT3_1 f_BIT2_0
f_BIT0_4 0 0 0 sim_or f_BIT1_0 f_BIT1_3
f_BIT1_4 0 0 0 sim_and f_BIT2_1 f_BIT0_3
f_BIT2_4 0 0 0 sim_or f_BIT3_0 f_BIT3_3
f_BIT3_4 0 0 0 sim_and f_BIT2_3 f_BIT0_3
f_BIT0_5 0 0 0 sim_and f_BIT2_1 f_BIT0_4
f_BIT1_5 0 0 0 sim_or f_BIT2_0 f_BIT0_5
f_BIT2_5 0 0 0 sim_and f_BIT2_3 f_BIT0_4
f_BIT3_5 0 0 0 sim_or f_BIT2_4 f_BIT2_5
f_BIT0_6 0 0 0 sim_and f_BIT0_1 f_BIT4_7
f_BIT1_6 0 0 0 sim_and f_BIT0_3 f_BIT4_7
f_BIT2_6 0 0 0 sim_and f_BIT1_4 f_BIT4_7
f_BIT3_6 0 0 0 sim_and f_BIT3_4 f_BIT4_7
f_BIT0_7 0 0 0 sim_or f_BIT0_0 f_BIT0_6
f_BIT1_7 0 0 0 sim_or f_BIT0_4 f_BIT1_6
f_BIT2_7 0 0 0 sim_or f_BIT1_5 f_BIT2_6
f_BIT3_7 0 0 0 sim_or f_BIT3_5 f_BIT3_6
s_BIT0_8 0 0 0 sim_exor f_BIT0_2 f_BIT4_7
s_BIT1_8 0 0 0 sim_exor f_BIT1_2 f_BIT0_7
s_BIT2_8 0 0 0 sim_exor f_BIT2_2 f_BIT1_7
s_BIT3_8 0 0 0 sim_exor f_BIT3_2 f_BIT2_7
```

multiplier_tb_template

```
//--------------------------------------------------------------------------------
// version 1.1 July 24,2023
//--------------------------------------------------------------------------------
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <math.h>
using namespace std;

int  sim_ha_cout(int a,int b,int delay);
int  sim_ha_sout(int a,int b,int delay);
int  sim_fa_cout(int a,int b,int c,int delay);
int  sim_fa_sout(int a,int b,int c,int delay);
int  sim_and(int a,int b,int delay);
int  sim_or(int a,int b,int delay);
int  sim_exor(int a,int b,int delay);
int  sim_wire0(int a,int delay);
int  sim_wire1(int a,int delay);
int  sim_inv(int a,int delay);
void netlist_BITSIZE0xBITSIZE0();
void netlist(int *a,int *b,int *s,int delay);

int main(int argc, char * const argv[]) {
  netlist_BITSIZE0xBITSIZE0();
  return 0;
}

//-------------------------------------------
//function : BITSIZE0xBITSIZE0
//-------------------------------------------
void netlist_BITSIZE0xBITSIZE0(){
  int i,j,k;
  int loopnum;
  int delay;

  int a[BITSIZE0];
  int b[BITSIZE0];
  int s[BITSIZE0*2];
  long int atmp,btmp;
  long int ain,bin,sout,sout2;
  long int bitpower;

  loopnum=10000;

  bitpower=1;
  for (k=0; k<BITSIZE0; k++) {
    bitpower=bitpower*2;
  }

  delay=0;
  unsigned int now=(unsigned int)time(NULL);
  srand(now);
  for (i=0; i<(loopnum+1); i++) {
    if(i==loopnum){
      delay=1;
      ain=0;
      bin=0;
    }
    else{
      //input
      ain=rand()%bitpower;
      bin=rand()%bitpower;
      sout=ain*bin;
```

```
    }

    //bit epansion
    atmp=ain;
    btmp=bin;
    for (k=0; k<BITSIZE0; k++) {
      a[k]=atmp%2;
      atmp=atmp/2;
      b[k]=btmp%2;
      btmp=btmp/2;
    }

    //-----------------
    // netlist body
    //-----------------
    netlist(a,b,s,delay);

    if(i==loopnum){
      //delay
      for (k=0; k<BITSIZE0*2; k++) {
        if(k%4==0){
          printf("sum delay s[%d]:%d ",k,s[k]);
        }
        else if(k%4==3){
          printf("s[%d]:%d \n",k,s[k]);
        }
        else{
          printf("s[%d]:%d ",k,s[k]);
        }
      }
      printf("\n");
    }
    else{
      //logic
      sout2=0;
      for (k=(BITSIZE0*2-1); k>=0; k--) {
        sout2=2*sout2+s[k];
      }
      printf("   ain: %3ld  bin: %3ld  sout: %3ld    sout2: %3ld",ain,bin,sout,sout2);
      if(sout==sout2){printf("   allPASS\n");}
      else{printf("   allFAIL\n");}
    }
  }
}

int sim_and(int a,int b,int delay){
  int y;
  if(delay==0){
    y=a&b;
    return y;
  }
  else{
    if(a>=b){y=a+1000;}
    else{y=b+1000;}
    return y;
  }
}
int sim_or(int a,int b,int delay){
  int y;
  if(delay==0){
    y=a|b;
    return y;
  }
  else{
    if(a>=b){y=a+1000;}
    else{y=b+1000;}
```

```
    return y;
  }
}
int sim_wire0(int a,int delay){
  int y;
  if(delay==0){
    y=a;
    return y;
  }
  else{
    y=a;
    return y;
  }
}
int sim_wire1(int a,int delay){
  int y;
  if(delay==0){
    y=a;
    return y;
  }
  else{
    y=a;
    return y;
  }
}
int sim_inv(int a,int delay){
  int y;
  if(delay==0){
    y=~a;
    return y;
  }
  else{
    y=a+1000;
    return y;
  }
}
int sim_exor(int a,int b,int delay){
  int y;
  int n0,n1,n2;
  n0=sim_or(a, b,delay);
  n1=sim_and(a, b,delay);
  n2=sim_inv(n1,delay);
  y=sim_and(n0, n2,delay);
  return y;
}
int sim_ha_cout(int a,int b,int delay){
  int y;
  y=sim_and(a, b,delay);
  return y;
}
int sim_ha_sout(int a,int b,int delay){
  int y;
  int n0,n1;
  y=sim_exor(a,b,delay);
  return y;
}
int sim_fa_cout(int a,int b,int c,int delay){
  int y;
  int n0,n1,n2;
  n0=sim_and(c, b,delay);
  n1=sim_or(c, b,delay);
  n2=sim_and(n1, a,delay);
  y=sim_or(n0, n2,delay);
  return y;
}
int sim_fa_sout(int a,int b,int c,int delay){
```

```
  int y;
  int n0,n1;
  n0=sim_exor(c,b,delay);
  y=sim_exor(n0,a,delay);
  return y;
}
```

layout_template

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">

//-------------------------------------------------------------------------------
// version 1.1 July 24,2023
//-------------------------------------------------------------------------------

window.onload = function() {
  draw();
};

function draw() {
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");

ctx.fillStyle = "#000000";
ctx.fillRect(0,0,1400,3000);

ctx.font = '14px Helvetica';
//AND gate
ctx.fillStyle = "#880000";
ctx.fillRect(20,20,15,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('AND gate', 50, 35);
//OR gate
ctx.fillStyle = "#008800";
ctx.fillRect(140,20,15,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('OR gate', 170, 35);
//EXOR gate
ctx.fillStyle = "#000088";
ctx.fillRect(260,20,15,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('EXOR gate',290, 35);
//half adder
ctx.fillStyle = "#888800";
ctx.fillRect(400,20,30,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('half adder', 450, 35);
//full adder
ctx.fillStyle = "#880088";
ctx.fillRect(550,20,30,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('full adder', 600, 35);
//input
ctx.fillStyle = "#888888";
ctx.fillRect(20,60,15,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('input a,b', 50, 75);
//wire
ctx.fillStyle = "#008888";
ctx.fillRect(170,60,15,15);
ctx.fillStyle = "#ffffff";
ctx.fillText('wire(no logic of tree)', 200, 75);
//cla4b
ctx.beginPath();
ctx.moveTo(380,60);
ctx.lineTo(400,60);
ctx.lineTo(400,80);
ctx.lineTo(380,80);
ctx.lineTo(380,60);
ctx.strokeStyle="white";
```

```
ctx.lineWidth=1;
ctx.stroke();
ctx.beginPath();
ctx.fillStyle = "#ffffff";
ctx.fillText('cla4b', 410, 75);

//bit number
ctx.font = '14px Helvetica';
ctx.fillStyle = "#ffffff";
ctx.fillText('0',   2,230);
ctx.fillText('1',  22,230);
ctx.fillText('2',  42,230);
ctx.fillText('3',  62,230);
ctx.fillText('4',  82,230);
ctx.fillText('5', 102,230);
ctx.fillText('6', 122,230);
ctx.fillText('7', 142,230);
ctx.fillText('8', 162,230);
ctx.fillText('9', 182,230);
ctx.fillText('10',202,230);
ctx.fillText('11',222,230);
ctx.fillText('12',242,230);
ctx.fillText('13',262,230);
ctx.fillText('14',282,230);
ctx.fillText('15',302,230);
ctx.fillText('16',322,230);
ctx.fillText('17',342,230);
ctx.fillText('18',362,230);
ctx.fillText('19',382,230);
ctx.fillText('20',402,230);
ctx.fillText('21',422,230);
ctx.fillText('22',442,230);
ctx.fillText('23',462,230);
ctx.fillText('24',482,230);
ctx.fillText('25',502,230);
ctx.fillText('26',522,230);
ctx.fillText('27',542,230);
ctx.fillText('28',562,230);
ctx.fillText('29',582,230);
ctx.fillText('30',602,230);
ctx.fillText('31',622,230);
// multiplier starts here.




ctx.fillStyle = "#000000";
ctx.fillRect(0,0,1,1);

}

</script>
</head>
<body>
 <canvas id="canvas" width="1400" height="3000"></canvas>
</body>

</html>
```